Trabalho I
Programação de Software Básico

Márcio Santos do Carmo[1], **André Luiz Marques Macrini Leite**[2]

[1]Politécnica – Pontifícia Universidade Católica do Rio Grande do Sul (PUCRS)
90.619-900 – Porto Alegre – RS – Brasil

marcio.carmo@acad.pucrs.br, andre.leite.001@acad.pucrs.br

Resumo. *Este artigo traz uma proposta de solução para o primeiro trabalho da disciplina de Programação de Software Básico, onde o objetivo é praticar a manipulação de memória em baixo nível, implementando uma técnica simples de esteganografia em imagens.*

1. Introdução

Esteganografia é um método bastante usado para ocultar informações. Ela consiste em aplicar técnicas capazes de ocultar informações dentro de outros arquivos como imagens, músicas, vídeos ou mesmo anúncios de jornais e revistas. Para o desenvolvimento deste trabalho, utilizaremos o conceito de esteganografia junto a uma técnica chamada Cifra de César, utilizando uma das cores dentre o RGB de um píxel escolhido algoritmicamente para esconder uma mensagem dentro de uma imagem. Essa mensagem não somente será escondida dentro da imagem, mas também estará protegida por uma senha, a qual será utilizada posteriormente para recuperar a mensagem que foi escondida. Para solucionar o problema proposto criamos dois projetos ”Encrypt” e ”Descrypt”, ambos na linguagem de programação C.

2. Encrypt

O projeto Encrypt foi criado com o propósito de pegar a mensagem que o usuário deseja criptografar, usar uma cifra para criptografa-la e esteganografar a mensagem criptografada em uma imagem. No projeto foram criados seis métodos para resolver o problema:

1. load
2. salta
3. encrypt
4. cifrar
5. passwordInput
6. messageInput

2.1. Load

Este método serve para carregar uma imagem para dentro do programa dentro de uma variável.

```
void load(char* name, Img* pic) {
    int chan;
    pic -> img = (unsigned char*) SOIL_load_image(name, &pic -> width, &pic -> height, &chan, SOIL_LOAD_RGB);
    if(!pic -> img) {
        printf( "SOIL loading error: '%s'\n", SOIL_last_result() );
        exit(1);
    }
}
```

2.2. Salta

O ”Salta” define o salto entre os pixels da imagem. Recebe por parâmetro a senha que foi digitada pelo usuário, então soma o valor decimal de todos os caracteres da senha. Após isso divide o resultado da soma pelo tamanho da senha. Esse valor é utilizado para espalhar os bits dos caracteres nos pixels da imagem.

```
void salta(char* password) {
    int salto = 0;
    for(int i = 0; i < strlen(password); i++) {
        salto += password[i]; // SOMA O VALOR DECIMAL DA TABELA ASCII PARA O CARACTER DA SENHA NA ITERAÇÃO
    }
    salto /= strlen(password);
    cont += salto;
}
```

2.3. Cifrar

Para criptografar a mensagem tomamos como método a cifra de Cesar. Para utilizar a cifra, criamos o método ”Cifrar” que recebe a mensagem por parâmetro e devolve ela criptografada. A cifra é realizada com base no tamanho da mensagem. A cifra obtém o tamanho da mensagem dividido por dois e soma com o valor decimal de cada caractere da mensagem. Depois retorna a mensagem criptogradada.

```
char* cifrar(char* message){
    int tam = strlen(message);
    int cesar = (tam/2);
    char* cifra = calloc(300, sizeof cifra);
    for(int i=0; i<tam; i++){
        cifra[i] = message[i] + cesar;
    }
    return cifra;
}
```

2.4. Encrypt

Tem como objetivo criptografar a mensagem recebida por parâmetro e esteganografar a mensagem já criptografada em uma imagem. Para esteganografia é utilizado o métoo ”Salta”, mencionado anteriormente. Primeiramente o ”Encrypt” chama o método ”Cifrar”, para criptografar a mensagem, passando por parâmetro a própria mensagem que o usuário digitou. Então utiliza o operador bitwise Left Shift para empurrar os bits de cada pixel RED uma vez para esquerda. Depois faz uma operação de OR para colocar no bit menos significativo do pixel RED os bits de cada letra. Os bits das letras são espalhados pelos pixels da imagem por conta do método ”Salta” que é chamado cada vez que um bit é gravado em um pixel.

```
for(int j = 0; j < 8; j++) {
    switch(j) {
        case 0:
            pic.img[cont].r = (pic.img[cont].r << 1) | bit8;
            salta(password);
            break;
        case 1:
            pic.img[cont].r = (pic.img[cont].r << 1) | bit7;
            salta(password);
            break;
        case 2:
            pic.img[cont].r = (pic.img[cont].r << 1) | bit6;
            salta(password);
            break;
        case 3:
            pic.img[cont].r = (pic.img[cont].r << 1) | bit5;
            salta(password);
            break;
        case 4:
            pic.img[cont].r = (pic.img[cont].r << 1) | bit4;
            salta(password);
            break;
        case 5:
            pic.img[cont].r = (pic.img[cont].r << 1) | bit3;
            salta(password);
            break;
        case 6:
            pic.img[cont].r = (pic.img[cont].r << 1) | bit2;
            salta(password);
            break;
        case 7:
            pic.img[cont].r = (pic.img[cont].r << 1) | bit1;
            salta(password);
            break;
```

Após esteganogradar a mensagem inteira na imagem, é marcado o próximo pixel com um caractere especial para delimitar o fim da mensagem.

```
cont++;
pic.img[cont].r = 0x23;
```

2.5. PasswordInput

Solicita para o usuário a senha para criptografar e espalhar a mensagem pela imagem.Foi utilizada a lib ¡conio.h¿ para que a senha digitada fosse alterada por ”*” na tela do console.

```
char* passwordInput() {
    char* password = calloc(50, sizeof password);
    printf("Senha:", password);
    int i;
    fflush(stdin);
    for (i = 0; i < 300; i++)
    {
        password[i] = getch();
        if(password[i] == 13){
            break;
        }
        putchar('*');
    }
    printf("\n");
    password[i] = '\0';
    return password;
}
```

2.6. MessageInput

Solicita para o usuário a mensagem a ser criptografada e esteganografada na imagem.

```
char* messageInput() {
    char* message = calloc(300, sizeof message);
    printf("Mensagem:", message);
    gets(message);
    return message;
}
```

3. Decrypt

O projeto Decrypt foi criado com o propósito de carregar a imagem já estenografada e a partir da senha dada pelo usuário extrair a mensagem criptografada da imagem. Após isso, o programa deve descriptografar a mensagem e exibi-la no console. No projeto foram criados cinco métodos para resolver o problema:

1. load
2. salta
3. decrypt
4. decifrar
5. passwordInput

3.1. Load

Este método serve para carregar uma imagem para dentro do programa dentro de uma variável.

```
void load(char* name, Img* pic) {
    int chan;
    pic -> img = (unsigned char*) SOIL_load_image(name, &pic -> width, &pic -> height, &chan, SOIL_LOAD_RGB);
    if(!pic -> img) {
        printf( "SOIL loading error: '%s'\n", SOIL_last_result() );
        exit(1);
    }
}
```

3.2. Salta

Assim como o método do projeto Encrypt, o Salta define o salto entre os pixels da imagem. Recebe por parâmetro a senha que foi digitada pelo usuário, então soma o valor decimal de todos os caracteres da senha. Após isso divide o resultado da soma pelo tamanho da senha. Esse valor é utilizado para espalhar os bits dos caracteres nos pixels da imagem.

```
void salta(char* password) {
    int salto = 0;
    for(int i = 0; i < strlen(password); i++) {
        salto += password[i]; // SOMA O VALOR DECIMAL DA TABELA ASCII PARA O CARACTER DA SENHA NA ITERAÇÃO
    }
    salto /= strlen(password);
    cont += salto;
}
```

3.3. Decifrar

Realiza o procedimento oposto do método ”Cifrar”, presente no projeto Encrypt. Criamos o método Decifrar que recebe a mensagem criptografada por parâmetro e devolve ela descriptografada. O método obtém o tamanho da mensagem criptografada dividido por dois e subtrai com o valor decimal de cada caractere da mensagem. Depois retorna a mensagem descriptogradada.

```
char* decifrar(char* message){
    int tam = strlen(message);
    int cesar = (tam/2);
    char* cifra = calloc(300, sizeof cifra);
    for(int i=0; i<tam; i++){
        cifra[i] = message[i] - cesar;
    }
    return cifra;
}
```

3.4. Decrypt

Tem como objetivo descriptografar a mensagem criptografada que está esteganografada na imagem. Este método faz uma varredura nos pixels da imagem buscando os bits gravados no RED. Ele busca de pixel em pixel utilizando o método salta para pegar o próximo pixel até encontrar a marcação do fnal da mensagem. Utiliza o operador bitwise Right Shift em cada pixel que passa para ober o bit menos significativo e armazena o mesmo nas variáaveis bit1-bit8. Quando encontra oito bits, o método concatena os bits encontrados, formando um caractere e armazena o caractere na variável mensagemCriptografada.

```
for(int i = 0; i < pic.height*pic.width; i++) {
    if(pic.img[cont+1].r == 0x23){break;}
    bit8 = (pic.img[cont].r >> 0)  & 0x01;
    salta(password);
    bit7 = (pic.img[cont].r >> 0) & 0x01;
    salta(password);
    bit6 = (pic.img[cont].r >> 0) & 0x01;
    salta(password);
    bit5 = (pic.img[cont].r >> 0) & 0x01;
    salta(password);
    bit4 = (pic.img[cont].r >> 0) & 0x01;
    salta(password);
    bit3 = (pic.img[cont].r >> 0) & 0x01;
    salta(password);
    bit2 = (pic.img[cont].r >> 0) & 0x01;
    salta(password);
    bit1 = (pic.img[cont].r >> 0) & 0x01;
    salta(password);

    messageCriptografada[i] = (bit8 << 7) + (bit7 << 6) + (bit6 << 5) + (bit5 << 4) + (bit4 << 3) + (bit3 << 2) + (bit2 << 1) + (bit1 << 0);
}
```

Após obter a mensagem criptografada, é chamado o método ’Decifrar”, e por fim, mostra a mensagem descriptografada no console.

3.5. PasswordInput

Assim como no método do projeto Encrypt, solicita para o usuário a senha que foi utilizada para criptografar e espalhar a mensagem pela imagem, porém desta vez a senha srá utilizada para buscar os caracteres espalhados pela imagem e descriptografar a mensagem. Foi utilizada uma lib para que a senha digitada fosse alterada por ”*” na tela do console.

```
char* passwordInput() {
    char* password = calloc(50, sizeof password);
    printf("Senha:", password);
    int i;
    fflush(stdin);
    for (i = 0; i < 300; i++)
    {
        password[i] = getch();
        if(password[i] == 13){
            break;
        }
        putchar('*');
    }
    printf("\n");
    password[i] = '\0';
    return password;
}
```